Scramble Suit: A Profile Differentiation Countermeasure to Prevent Template Attacks

Alessandro Barenghi, William Fornaciari, *Senior Member, IEEE,*
Gerardo Pelosi, *Member, IEEE,* and Davide Zoni

***Abstract*—Ensuring protection against side channel attacks is a crucial requirement in the design of modern secure embedded systems. Profiled side channel attacks, the class to which template attacks and machine learning attacks belong, derive a model of the side channel behavior of a device identical to the target one, and exploit the said model to extract the key from the target, under the hypothesis that the side channel behaviors of the two devices match. We propose an architectural countermeasure against cross-device profiled attacks which differentiates the side-channel behavior of different instances of the same hardware design, preventing the reuse of a model derived on a device other than the target one. In particular, we describe an instance of our solution providing a protected hardware implementation of the AES block cipher and experimentally validate its resistance against both Bayesian templates and machine learning approaches based on support vector machines also considering different state of the art feature reduction techniques to increase the effectiveness of the profiled attacks. Results show that our countermeasure foils the key retrieval attempts via profiled attacks ensuring a key derivation accuracy equivalent to a random guess.**

***Index Terms*—Embedded Systems Security, Side Channel Attacks Countermeasures, Profiled Attacks, Applied Cryptography**

I. Introduction

Modern embedded systems are pervasively deployed in our living environment, and they are increasingly in charge of performing sensitive tasks such as environmental parameter monitoring and mechanical actuation. In particular, the steady rise of the number of interconnected computing devices deployed in every aspect of human activities, commonly known as the Internet of Things (IoT), calls for a particular attention to their security requirements. Indeed, in the typical IoT scenario, such embedded computing systems are required to meet both stringent energy envelopes, due to their untethered functioning, and sound security guarantees, to avoid potential damage to the physical systems they may monitor or actuate [1]. Such requirements are typically fulfilled with the use of effective and efficient cryptographic primitives, implemented as either software libraries or dedicated hardware accelerators.

One of the crucial aspects of providing security guarantees in the IoT scenario is the possibility for an attacker to seize and tamper with the device, due to its embedded nature and field deployment. As a consequence, Side Channel Attacks (SCAs) have become one of the prime security threats to modern IoT devices, as they are applicable even in cases where correct and standard abiding implementations of cryptographic primitives are being used to provide the required security guarantees.

SCAs exploit the intrinsic link between the data being processed by a digital computing platform and one or more environmental parameters of the platform itself, such as the power consumption, Electro-Magnetic (EM) radiations or computing time [2]–[7]. Subsequently, given the measurements, the attacker tries to deduce the correct value of the secret key by modeling a small portion of the computation (e.g., a single instruction on a micro-controller), which in turn depends only on a small portion of the secret key. The assumption that the behavior of a small portion of the computing logic depends only on a few secret key bits allows the attacker to derive a model of the computing logic behavior for each possible value assumed by the key bits. The said models are compared with the actual measurements revealing the best fitting one, and thus the actual secret key value. Since the side channel measurements are affected by both random and systematic noise, the goodness of fit of a model to the actual device behavior is performed employing statistical tests, over a significant amount of side channel samples.

Countermeasures against SCAs rely on lowering the signal-to-noise ratio, a technique known as *hiding* [3], [8], [9], on computing the cryptographic algorithm in a semantically equivalent fashion, while randomizing the individual portions of the computation itself, a technique known as *masking* [3], [10], or by changing continuously the code employed to perform the sensitive computation, a technique known as *morphing* [11]–[15]. A further approach relies on trying to reduce the imbalance between two instances of the computing device, which act on Boolean complementary data, and are thus expected to exhibit an overall consumption which is constant [16]. All these approaches raise the required amount of measurements to be taken to obtain a sound statistical comparison among the key-dependent models, up to the point where either collecting the measurements, or processing the collected data exceeds practical feasibility. Indeed, it is not uncommon to evaluate the robustness of a SCA countermeasure in terms of the amount of Measurements To Disclose the key (MTD) [17], [18]. A crucial point in applying SCA countermeasures to IoT devices is their required overhead in terms of energy consumption and

A. Barenghi, W. Fornaciari, G. Pelosi (corresponding author), and D. Zoni were with the Department of Electronics Information and Bioengineering – DEIB, Politecnico di Milano, Milano, Italy.
E-mail: alessandro.barenghi@polimi.it, william.fornaciari@polimi.it, gerardo.pelosi@polimi.it, davide.zoni@polimi.it

Manuscript received Dec. 21, 2018; first revision Apr. 5, 2019; accepted May 24 2019. Date of publication xxx. xx xxxx.

execution time, which is common to be one to two orders of magnitude larger than the computation of an unprotected cipher [10]. To this end, a viable approach to prevent SCAs from succeeding is to employ a persistent memory counter in the device, which limits the maximum amount of cipher executions to one which is both compatible with the device lifetime and lower than the MTDs provided by hiding or masking countermeasures [19].

While this strategy is really effective in preventing common SCAs, a set of strategies, known as *profiled attacks*, was devised with the purpose of attacking a target with a minimal amount of measurements, down to a single one in most cases. Since the pioneering work on the Bayesian Template Attack (a.k.a. *Template Attack*) [19], profiled attacks have proved to be successful in attacking embedded devices and currently have no countermeasure explicitly aimed at foiling them. Profiled attacks involve collecting side channel measurements from another instance of the device under attack, on which the attacker may set the secret key value at his will, and perform as many measurements as needed (e.g., a blank smart card on which the private key can be set at the attacker's will). This allows him to derive a side channel behavior, or *profile*, which is a perfect match of the device under attack for every value which can be taken by the few secret key bits which are being retrieved at each attack. The approach proposed in [19] was extended and improved employing well-known statistical classifiers [20]–[22], which perform the same role of the maximum likelihood classification step done in template attacks, and provide an alternative way to obtain robustness against measurement noise in the key recovery phase [22]. Template attacks are indeed able to reach the maximum information theoretical effectiveness in deriving a secret key from a side channel behavior [19], [23]–[25], and are able in principle to defeat any of the countermeasures devised for non-profiled attacks, such as masking and hiding, provided that enough measurements are collected [26]. To this end, [22], [27] report that countermeasures effective against non-profiled side channel attacks are not sufficient to ward off against profiled ones. This has lead to practical attacks against real world devices [28], despite some may have very tight limitations on the number of collectible measurements [29]. A fundamental assumption of profiled attacks is that the side channel behavior derived on an instance of a device provides a good, ideally perfect, fit for the behavior of another instance of the same device. While this is correct in principle, process variability was shown to be of some hindrance to the creation of a correct profile from a single instance of a device [30], [31], although the authors of [30], [31] show that some tolerance to it can be achieved computing a synthetic device profile combining the measurements obtained from multiple instances of the same device. Finally, we note that, to the best of the authors' knowledge, no SCA countermeasure tailored explicitly against profiled attack exists.

Contributions. In this work we propose Scramble Suit, a countermeasure against profiled-SCAs employing either the power consumption or EM emission as the information leaking side channel. Our approach disrupts the possibility of reusing a profile acquired on a given device on a different one, whilst it does not prevent the acquisition of the profile itself. We propose an architecture-level approach which differentiates the side channel profiles of each single chip instance of the given design, without resorting to direct manipulation of low level design features or requiring a specific electronic design automation (EDA) toolchain. To this end, in the computation of the cryptographic primitive we employ data obtained as the response of a Physically Unclonable Function (PUF) to an input challenge, in such a way that it is not possible to separate its contribution from the side channel profile. Such an approach augments the natural effect of process variability to the point that reusing a side channel profile is not feasible, while preserving the benefits of the low variability coming from an optimal design. As a case study we describe a protected hardware module for the Advanced Encryption Standard (AES) cipher, and we practically validate its robustness instantiating the design on an FPGA platform, attempting to perform profiled side channel attacks against it with both the classic template attack approach and the support vector machine (SVM) classifier. In both cases, the cross-device prediction accuracy of each single-bit key value is reduced to 50% in the AES implementation protected with Scramble Suit. The results substantiate the fact that the resistance to profiled attacks is not dependent on a specific choice of the attack strategy.

We note that, since the Scramble Suit countermeasure acts preventing only the reuse of the information concerning the side channel profile of a device, non-profiled side channel attacks will still be possible on a device protected with the proposed countermeasure alone. To cope with this issue, Scramble Suit can be naturally combined with any of the countermeasures effective against non-profiled attacks (e.g., the ones in [32]). Indeed, such a countermeasure combination fits the fact that, if no countermeasures are present, a non-profiled is the simplest approach. Therefore, if protection against template attacks is taken into consideration, all the other more exploitable threats should have been dealt with already. In this work we do not tackle the topic of choosing a specific PUF architecture: Scramble Suit can be instantiated with any sound PUF instance [33]–[35].

The remainder of this work is organized as follows: Section II provides an overview on profiled attacks and summarizes the basic concepts on PUFs. Section III presents the Scramble Suit approach, detailing its working principle, and describing a protected AES implementation. Section IV reports the results of our experimental validation and Section V summarizes our conclusions.

II. Background

In this section we provide an overview on the techniques to lead a profiled side channel attack, and a summary of the fundamental properties of PUFs.

A. *Profiled Side Channel Attacks*

Profiled SCAs use the information unintentionally transmitted on side channels by digital computing devices proceeding in two subsequent phases: a profiling phase and an exploitation phase. Their most significant advantage lies on the ability to recover the cryptographic secret key employed by the targeted cryptographic primitive implementation with a few (in some cases only a single) side channel measurement from the executing device, overcoming implementations or countermeasures which restrict the number of measurements that can be acquired.

The profiling phase of this cross-device attack may be interpreted as an instance of a supervised learning problem, where the ground truth required to train a classifier is obtained from a device fully controlled by the attacker, so that the side channel behavior can be matched to a known key value. In particular, in the profiling phase the attacker derives an accurate model of the side channel information leakage (e.g., via power consumption, or EM radiation) employing a different instance of the device he wants to retrieve the key from. The instance employed by the attacker differs from the target one only for the possibility of controlling the value of the employed key [36], [37]. We note that such an instance is legitimately obtained and is not an unauthorized or modified clone of the original chip design. The model learned in the training phase is used, in the exploitation phase, to obtain the unknown key of the target device via classification of a few measurements from it.

Profiled SCAs share with their classic counterpart the divide-et-impera approach which retrieves the secret key one small portion at a time. In this respect, considering the key as split up into bit- or byte-sized parts, the profiling phase builds a set of side channel behavior profiles for each key portion. Each one of these sets contains the models for the behavior of the device for any possible value that the aforementioned small key portion may take.

Given the assumption that the profile is correctly derived, a profiled-SCA can be viewed as the most effective attack in an information theoretic sense [19]. As a consequence, the use of the profiled attack methodology was also suggested to evaluate the side channel resistance of a cryptographic implementation in a worst case scenario [30]. However, the extent to which a leakage model that is carefully obtained for one device can be used to lead successful attacks against another instance of the same device may be diminished by either outdated profiling data [38] or process variability issues due to technology scaling effects. In the former case, [38] showed that the problem can be effectively managed assuming the measurements used in the exploitation phase as being both desynchronized and differently scaled in amplitude with respect to the data used in the profiling phase. In the latter case, [31] showed that with a 65nm 8-bit datapath AES implementation the process variability effects can be compensated building a leakage model employing the profiling data of multiple chips. This implies that the *distance* between the profiles of the same key value on two different chips is smaller than the distance between the profiles of two different key value on the same chip. If that is not the case, i.e., the behaviors of a device under two different values of a key portion are closer than the ones of two device instances under the same value of the key portion, the side channel profiles obtained from the measurements of a multiple number of devices will not be able to perform a successful key recovery in the exploitation phase of the attack. Indeed, in the latter case, any strategy aimed at classifying the behavior of a device with a model taken with multiple instances would lack the selectivity required to tell apart the key dependent behaviors.

In this work, we will validate the effectiveness of our countermeasure against both the classical template attacks (TAs), as first proposed in [19], and an instance of the more recent approaches performing profiled-SCAs using machine learning techniques. Indeed, machine learning techniques have been employed to perform profiled attacks, building the model that best describes the behavior of the targeted device in a non-parametric and data-driven way, thus removing any assumption on the statistical distribution of the information leakage. In particular, following the pioneering work of Lerman et al. [39], the supervised learning approach employed by Support Vector Machines (SVMs) has proved to be an interesting alternative to TAs [20]–[22], and is thus employed as a second benchmark of the robustness of our countermeasure. Finally, in template attacks as well as in machine learning profiled SCAs, the large number of samples in a sequence of side channel measurement (a.k.a. *trace*) may prove a hindrance to the effectiveness of the attacks due to numerical instability and the high amount of information required to estimate the statistical distribution of the behaviors of the device. To this end, feature selection techniques have been successfully applied, increasing significantly the effectiveness of profiled-SCAs. In the following we provide a summary of the background on template attacks, SVM-based profiled attacks and feature selection techniques to enhance their effectiveness.

Template Attacks. Assuming a cryptographic primitive implementation fed with an l-bit key k (having a value chosen by the attacker) and a number $n{\gg}1$ of input plaintexts, the sequence of side channel measurements obtained from the running device, for each single-bit key value k_i, $1{\le}i{\le}l$, over a time interval of length s, is commonly named as *trace* and denoted as $\hat{T}^{(k_i)}{=}\{\hat{T}_j(t)\,|\,1{\le}t{\le}s\},\ 1{\le}j{\le}n$.

Template attacks assume that each side channel measurement is affected by additive Gaussian noise, and model each trace as a random vector variable $T^{(k_i)}$ following a multivariate Gaussian distribution with mean vector $\mu^{(k_i)}$, and covariance matrix $\Sigma^{(k_i)}$, i.e.: $T^{(k_i)}{\sim}\mathcal{N}(\mu^{(k_i)},\Sigma^{(k_i)})$, with probability density function

$$\Pr(T^{(k_i)}=x)=\frac{\exp\big(-\frac{1}{2}(x-\mu^{(k_i)})(\Sigma^{(k_i)})^{-1}(x-\mu^{(k_i)})^{\mathrm{tr}}\big)}{\sqrt{(2\pi)^n\det(\Sigma^{(k_i)})}}.$$

In the profiling phase, the attacker obtains a sample estimate of the mean vector $\hat{\mu}^{(k_i)}$ and the covariance matrix $\hat{\Sigma}^{(k_i)}$, collecting sets of measurements for each value of the key bits k_i, $1{\le}i{\le}l$. The attacker thus derives the sample distributions of variables $T^{(k_i)}$, for each possible value of k_i: if the side channel profile associated to the single i-th bit of the secret key is considered, the attacker will derive two distributions for $T^{(k_i)}$ by setting a value for k_i (i.e., $k_i{=}0$ and $k_i{=}1$) and collecting the measurements from the controlled device.

In the exploitation phase, a trace $\hat{T}{=}\{\hat{T}(t)\,|\,1{\le}t{\le}s\}$ is acquired from the device under attack, for which the key value is unknown, employing the same measurements conditions of the profiling phase. The likelihood of $\hat{T}$ being an instance (i.e., sample) of one of the random vector variables $T^{(k_i)}{=}\{T(t)\,|\,1{\leq}t{\leq}s\}$, is evaluated as the a-posteriori probability:

$$\Pr(k_i\,|\,\hat{T}) = \frac{\Pr(T^{(k_i)} = \hat{T})\cdot\Pr(k_i)}{\sum_{h=1}^{l}\Pr(T^{(k_h)} = \hat{T})\cdot\Pr(k_h)}$$

where $\Pr(k_i)$ is the a-priori probability associated to the specific key-bit value k_i that does not consider $\hat{T}$. Typically, all key values are equally likely and hence $\Pr(k_i){=}\frac{1}{2}$. For each key-bit $1{\leq}i{\leq}l$, the value k_i (i.e., either $k_i{=}0$ or $k_i{=}1$) which maximizes the aforementioned a-posteriori probability is selected as the value of the i-th bit of the secret key employed by the device under attack. The choice of modeling the probability density functions of the $T^{(k_i)}$ variables as multivariate Gaussian functions has proven, in practice, to be a very good approximation of the side channel leakage on the power consumption channel, although other parametric distributions can be considered [40].

Support Vector Machines. Machine learning techniques replace the assessment of the maximum a-posteriori probabilities of each key-bit value, $\Pr(k_i\,|\,\hat{T})$, with a classification step able to guess the correct value of k_i. The profiling phase of the employed machine learning technique makes use of a set of traces $\{\hat{T}_j^{(k_i)}\}$, $\hat{T}_j^{(k_i)}{=}\{T_j(t)\,|\,1{\leq}t{\leq}s\}$, with $1{\leq}j{\leq}n$, each of which labeled as belonging to a group having the i-th bit of the key equal to zero or one, according to the actual value which was set in the attacker-controlled device during their gathering. Our classifier of choice, the SVM, enforces a non-probabilistic binary linear classifier assigning new (unlabeled) traces to one of these groups.

To the end of training the classifier, each trace $\hat{T}_j^{(k_i)}$ is mapped to a point of a multi-dimensional geometric space with the intent of determining clusters of data and dividing them into sets separated by clear boundaries. To this aim, a projection function $\phi(\cdot)$ mapping the original data to a space with higher (or infinite) dimensions is employed to convert complex boundaries into linear ones. In particular, the vector parameters w and b of an hyperplane, $w^{\mathrm{tr}}T{+}b$, that has the largest distance from the nearest data points of each set, is computed by formulating the following convex optimization problem: $\min\frac{1}{2}(w^{\mathrm{tr}}\cdot w)$, subject to $w^{\mathrm{tr}}\phi(\hat{T}^{(k_i)}){+}b{\geq}1$ if $k_i{=}1$, and $w^{\mathrm{tr}}\phi(\hat{T}^{(k_i)}){+}b{\leq}{-1}$ if $k_i{=}0$, for all the available traces $\hat{T}$. The traces which are projected onto points closest to the separating hyperplane are the ones which influence most its computation and are called *support vectors*, hence the name of the classifier. During the exploitation phase, traces collected from the device under attack are mapped into the same space and labeled as belonging to one set or the other, based on which side of the hyperplane they fall.

Feature selection. Most of the computational effort to perform a profiled-SCA lies in the management of traces with a large number of samples ($s{\gg}1$): cases where the number of samples of a trace are in the high hundreds or a few thousands are not uncommon in practice. In particular, naively discarding trace samples may cause poor results in terms of accuracy of the key recovery procedure, while employing too many of them will likely lead to the introduction of a higher amount of noise, potentially more than the amount of sensitive leakage information. A *feature selection procedure* is usually adopted to improve the efficiency and effectiveness of the entire procedure. Such a preprocessing step allows either to select a subset of the trace samples based on their sensitive leakage information content, or to filter out redundant and uncorrelated information present in the original dataset by combining the trace samples to obtain a smaller set of features. The most effective feature selection algorithms present in the state-of-the-art include: *i)* the selection of trace samples exhibiting maximum variance, *ii)* the selection of trace samples corresponding to the maximum Sum Of Squares of t-difference of means (SOST) [24], and *iii)* the linear combination of trace samples by means of the unsupervised Principal Component Analysis (PCA) [25], [41] as opposed to its supervised counterpart presented in [42]. While the first two feature selection approaches simply sort the time instants in a trace according to the goodness of their score, and consider only the subset of the best scoring ones, PCA takes a different approach to the feature reduction problem. Indeed PCA considers the samples of a trace as a set of interrelated random variables, and aims to reduce their number while retaining as much as possible the information included in their variation. This is done by transforming the original data into a new set of variables, their *principal components*, which are mutually uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original data.

B. Physically Unclonable Functions

Alongside common implementations of mathematically designed cryptographic primitives, a key role in secure circuits is played by the so called Physically Unclonable Functions (PUFs). PUFs are a hardware cryptographic primitive which, presented with a challenge bit-string c out of a set of possible ones C provides a response r which is expected to be unique for each device instance. It is thus possible to employ the response to a challenge as a unique fingerprint of a given device instance. According to [43], the fundamental characteristics of a PUF are: i) unclonability, i.e., a PUF instance should not be reproducible even if the attacker is supplied with all the design details of an existing one; ii) reliability, i.e., a given PUF instance should provide always the same response to a given challenge; iii) unpredictability, i.e., it should not be possible to guess a response to a given challenge. PUF designs are traditionally classified into two main categories: the so-called *weak* PUFs and the *strong* PUFs [44]. While a unique formal definition of a weak PUF is currently not agreed upon by the community, a widely accepted criterion is to define a weak PUF as a PUF where the number of admissible challenge-response pairs is polynomial in the number of elementary components (e.g., logic gates) composing the PUF [43]. By contrast, a strong PUF is a PUF where the number of admissible challenge-response pairs grows exponentially with the number of its elementary components. In this work, we will focus on strong PUFs, against which the main attack strategy is represented by trying to derive via machine learning algorithms their inner structure, so to be able to clone an instance of them.

In particular, machine learning attacks on PUFs rely on a significant amount of challenge-response pairs to estimate the values of the instance-dependent parameters of the PUF (e.g., line propagation delays). Once these parameters are obtained it is possible to build a circuit equivalent to the modeled PUF, thus breaking the unclonability property.

Concerning the application of side channel attacks to PUFs, recent work [45] has shown that it is possible to use the information coming from side channels to enhance the effectiveness of machine learning attacks to strong PUFs. In particular, observing the time required to produce a response and the power consumption of the PUF in doing so, it is possible to reduce the number of known challenge-response pair and attack most state-of-the-art PUFs, provided their inputs and outputs are known. In [46] the authors analyze from an active and passive side channel attack standpoint the *Controlled PUF* construction [47] relying on an arbiter PUF outputting a single bit per query, which is stored in a shift register. It is assumed that an attacker has no direct access to the challenges and responses. The authors of [46] show that, under the hypotheses of resetting the PUF and initializing the register for the storage of the responses to a fixed value at each query, it is possible to perform side channel attacks to such a design. Pushed by this avenue for attacks, a significant research effort was put into the design of PUFs resistant to modeling attacks which also exploit side channel information. Among the most recent works, the one proposed by Xi et al [33], reports a design of a side channel resistant, strong PUF, which was experimentally validated to be resistant. Such a design of a PUF would fit the requirements for an ASIC instantiation of the Scramble Suit countermeasure.

III. The Scramble Suit Approach

In this section we describe Scramble Suit, an architecture level approach against profiled-SCAs. Section III-A provides a high level overview of Scramble Suit, while an instance employing an AES-128 implementation as our case study is discussed in Section III-B.

A. High Level Structure

Counteracting profiled side channel attacks requires to invalidate one of the basic assumptions which allow them to succeed: *(1)* the attacker is able to derive the key-dependent side channel behavior (e.g., the power consumption profile) of a device he controls completely and *(2)* the attacker is able to reuse the profile of a device to extract information from an identical one, under the assumption that they share the same side channel behavior. Given that assumption *(1)* will always be fulfilled if an attacker collects enough measurements to obtain an accurate profile of the device, we focus on invalidating assumption *(2)* with Scramble Suit. Indeed, if different instances of the same device exhibit a substantially different side channel behavior arising from the processing of inputs and key, it is not possible for an attacker to employ a profile of a device he controls to extract information from a different one. Our countermeasure leverages a strong PUF [48] as a mean to amplify the effects of the process variability, and materialize them onto digital data which are fed to a computing circuit. However, the computation with the said data should provide a contribution to the side channel profile of the device which is not separable from the one of actual cipher computation. To this end, we propose to employ the data in a duplicated instance of the cipher itself, which is run in parallel with the one computing the actual primitive on real data. We note that it is not possible to substitute the PUF in our design with a generic True (i.e., physical) Random Number Generator (TRNG) since the digital data being fed to the computing circuit should be both uniquely bound to its structure, and reproducible. Indeed, the latter feature is required to systematically distort the side channel profile of the device, while TRNG generated inputs may be removed via averaging the data among different sets of measurements taken with the same secret key and a randomly re-generated fake one.

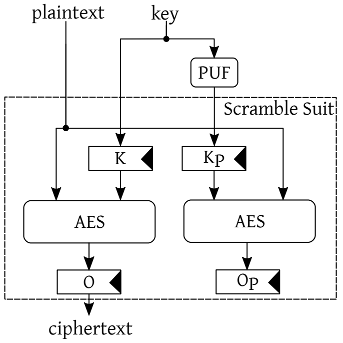


Figure 1. Conceptual view of Scramble Suit, employing an AES implementation as an exemplary case study

We note that, in case the cryptographic device is employing a single hard-coded key, it is possible to avoid the need of a PUF. Indeed, it is possible to generate off-line the value for the key to be employed by the replica cipher through a strong key derivation function, e.g., HKDF [49], fed with both the value of the actual device key and a manufacturer key which is never embedded into the device itself.

Figure 1 depicts a high level view of the method we propose to obtain a device dependent side channel behavior taking as an example a hardware implementation of the AES block cipher. The cipher key $\mathtt{K}$ is fed into a PUF, acting as a challenge for it, and the PUF response is employed as the device dependent key $\mathtt{K_P}$. Two instances of the AES cipher are fed with the same plaintext, and either $\mathtt{K}$ or $\mathtt{K_P}$, providing a contribution to the overall side channel behavior which depends both on the computed data and the device instance via $\mathtt{K_P}$. Under the assumption that it is not possible to collect measurements of each one of the AES instances present in Scramble Suit separately, either exploiting the spatial distribution of EM leakage, or distinguishing their execution in time, the device-dependent side channel behavior of the resulting protected design will depend on the key value in a different way for each device instance. The extent of such a difference is a consequence of two key properties of a strong PUF [48], [50], namely: *(1) unicity*, i.e., the property stating that two different instances of a PUF should output significantly different responses to the same challenge; *(2) uniformity*, which

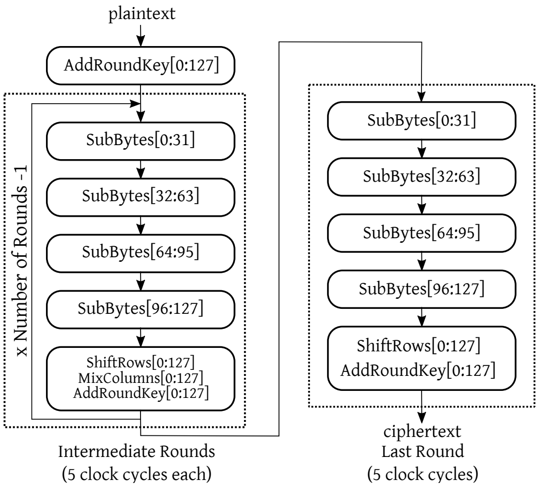


Figure 2. Data flow diagram of the AES-128 implementation employed as the baseline in our case study. The implementation at hand performs the computation of the SubBytes (i.e., the application of the function contained in the look-up table referred as S-Box) on four bytes per clock cycle, while the ShiftRows, MixColumns and AddRoundKey primitives are computed all in a single clock cycle by means of a single combinatorial module, on the entire state of the cipher

guarantees that small changes in a challenge will cause changes in the entire response. Such properties are crucial, along the *reliability* property (see Section II), i.e., the same PUF instance should always provide the same response to a given challenge. Properties *(1)* and *(2)* ensure that $\mathtt{K_P}$ is picked significantly far apart, in terms of Hamming distance, among different devices, and that small changes in the user key $\mathtt{K}$ will map to significant changes in $\mathtt{K_P}$. The former effect thus ensures that the added contribution to the side channel profile of the device is indeed device specific, while the latter ensures that it impacts significantly on it, even for changes of a small portion of the user key value. The *reliability* property ensures that the value of the device dependent key will be deterministically derived for each given user key/device instance pair. Thus, the contribution added to the side channel behavior by the cipher computation over the device dependent key $\mathtt{K_P}$ will not be removed if averaging over multiple measurements of the execution of the cipher on the same inputs.

We note that the per-device deterministic generation of the value of $\mathtt{K_P}$ from $\mathtt{K}$ via PUF prevents the attacker from profiling the side channel for all the possible pair of values of $\mathtt{K}, \mathtt{K_P}$, since they will not be generated by the PUF. For instance, taking the straw-man example of a single-bit $\mathtt{K}$ and a corresponding single bit $\mathtt{K_P}$ only two of the possible four pairs of values for $(\mathtt{K}, \mathtt{K_P})$ will be generated by the PUF on a given device.

In order to be effective, the contribution to the side channel profile provided by Scramble Suit should be more significant than the one employed by the classification strategy to correctly identify the value of the key portion k_i. Indeed, if this condition is not met, applying a profiling technique exploiting multiple devices to obtain a valid profile as described in [31] becomes applicable. Such a condition is typically met thanks to the *uniformity* of strong PUFs, which makes the value of $\mathtt{K_P}$ differ significantly whenever small changes take place in $\mathtt{K}$, in turn causing the contribution to the side channel to change to a comparatively large extent. However, if the said condition is not fulfilled employing a single additional cipher instance, the side channel contribution of the Scramble Suit countermeasure may be increased by adding a larger number of replicas of the primitive to protect. The keys for the aforementioned replicas may either be generated with a PUF having a different architecture, or obtained as the cryptographic hash of $\mathtt{K_P}$.

We note that machine learning attacks aimed at breaking the unclonability property of the PUFs, as the one proposed in [45], cannot be employed against Scramble Suit, since they require the attacker to be provided with both the side channel data and the challenge-response pairs of the PUF itself. However, the challenge-response pairs are not public in Scramble Suit; indeed the challenges are the secret keys.

B. Architectural Implementation

We now detail the design of a Scramble Suit protected AES block cipher. The AES block cipher is a 128-bit block cipher, of which we consider the version employing a 128-bit key, AES-128 from now on. AES-128 is composed of nine equal rounds where the cipher state is processed by four primitives, SubBytes, ShiftRows, MixColumns and AddRoundKey in the said order, and a final round where the MixColumns primitive is not applied. The 10 rounds are preceded by a single AddRoundKey performed on the plaintext itself.

Figure 2 reports the data flow diagram of the implementation we chose as our baseline for the case study (available in [51]). The said implementation computes the SubBytes primitive on four bytes of the AES state per cycle, resorting to four separate look-up tables (LUTs), and thus completing the computation of SubBytes in four cycles. The remaining three primitives (two, in the case of the last round) are computed by a combinatorial circuit in a single clock cycle, resulting in a five cycles per round latency.

In designing the Scramble Suit protected AES we need to ensure that our architecture meets the two assumptions of Scramble Suit, i.e., a sound PUF is employed, and the contribution to the side channel provided by the duplicated core is not separable via either spatial or temporal features. Concerning the first assumption, since the design of a sound PUF is out of scope for this work, we emulate its effects encrypting the user key $\mathtt{K}$ with AES-128 employing a fixed, device dependent key $\mathtt{K_{dev}}$ and consider the result of the encryption as our $\mathtt{K_P}$. The strong PUF properties are satisfied by this emulation due to the properties of the AES block cipher, in particular: the *unicity* requirement is satisfied thanks to the resistance of AES-128 to *related-key* attacks [52], the *uniformity* requirement is satisfied thanks to the immunity of AES to differential cryptanalysis [53], and the *reliability* property is met thanks to the deterministic nature of AES-128. We note that the generation of $\mathtt{K_P}$ is performed during the setup phase of the circuit and does not influence the side channel profile derivation. Such an assumption matches the fact that the structure of the ideal PUF should not be learned by means of side channel attacks.

To meet the second Scramble Suit assumption, our design should ensure that the two instances of the AES are indeed

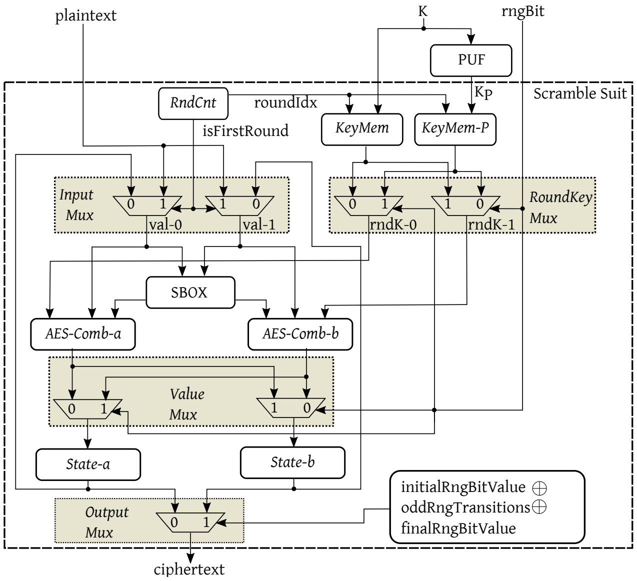


Figure 3. Architectural view of the Scramble Suit AES-128, reporting primary inputs and outputs. The AES-Comb-a and AES-Comb-b modules contain the combinatorial logic implementing the ShiftRows, MixColumns and AddRoundKey primitives, together with a bypass allowing the computation of the SubBytes primitive done by the SBOX module to be propagated to the state registers

run in perfect parallel, are identical one to the other, and are placed in such a way to prevent targeted attacks employing high spatial resolution EM probes [54]. Solving these issues without resorting to low level circuit design, including handmade placement and routing was proven to be difficult [54], [55], as EDA tools aggressively try to optimize the design in the "place and route" phase. Willing to retain the possibility of keeping the description of the entire design at register transfer level (RTL), we propose a design which randomly alternates the computation of the AES-128 employing K and the one employing $\mathrm{K_P}$ on two instances of the AES core. Employing this strategy allows us to foil attempts at distinguishing the contribution of one computation from the other, as they will both be performed an equal amount of times (on average) on each AES core. This, in turn, foils the attempts at employing spatial or temporal discrimination techniques. We note that there is no need to memorize the sequence of random values driving the choice of the core on which the computation is performed to be able to decrypt the obtained ciphertext, as it is identical to the one which is computed by an unprotected AES implementation.

Performing a trivial implementation of the mentioned strategy would incur in significant area penalties, due to the requirement of multiplexing data between the state registers holding the intermediate values of the computations with both K and $\mathrm{K_P}$ and the computation logic, plus the duplication of the round computation logic itself. We propose an optimized architecture for the Scramble Suit AES-128 reducing the amount of duplicated logic, and computing the cipher result in the same number of clock cycles as the baseline implementation. We note that the implementative approach we propose on the AES-128 case study can be applied in a straightforward fashion to any block cipher implementation. Moreover, it is also possible to generalize it to asymmetric primitives (encryption, signature) although it is required to analyze which are the datapaths in the cipher, to alternate the values belonging to the two computations.

Figure 3 depicts the architecture of our proposed Scramble Suit AES-128, while Figure 4 provides its timing diagram for the computation of the initial AddRoundKey and the first round of the cipher.

The Scramble Suit AES module has as its primary inputs the plaintext to be enciphered, the two keys K and $\mathrm{K_P}$ and the rngBit signal providing one random bit per AES-128 round. The module outputs the correct ciphertext after the encryption is complete. The two keys K and $\mathrm{K_P}$ are expanded by the KeyMem and KeyMem-P modules, which receive the round number r and perform the AES key schedule outputting the r-th round keys (namely, K^r $\mathrm{K_P}^r$ in the timing diagram in Figure 4). During the first cycle of the computation, the plaintext, from now on considered as a quadruple of 32-bit words $\langle P_0, P_1, P_2, P_3\rangle$ is duplicated by the Input Mux module, and stored in the two available state registers State-a and State-b after performing the initial AddRoundKey. The position of the device independent state $\langle I_0^0, I_1^I, I_2^0, I_3^0\rangle$ and the device dependent one $\langle D_0^0, D_1^0, D_2^0, D_3^0\rangle$ after the 0-th round only depends on the value of rngBit.

The Scramble Suit AES-128 alternates the computation of the AES rounds over the two available paths picking the storage location of the correct value at random between State-a and State-b at each round. This action, performed by the Value Mux pair of multiplexers, which employ rngBit as the selector, reduces the requirement for large muxes on the datapath to the minimum (a single pair for the states, plus a single pair for the round keys). Willing to reduce the area required by the protected datapath, we chose to instantiate a single LUT to compute the SubBytes primitive of the AES cipher (SBOX module in Figure 3). To avoid the increase in computation time mandated by the fact that the protected AES needs to compute the SubBytes primitive on twice the amount of data, we employ a dual-ported LUT to implement it, allowing the computations on both State-a and State-b simultaneously.

The choice of reducing the number of muxes to the minimum, while keeping the same latency for the entire encryption has the potential of causing a semantic mismatch in the computed values. Indeed, during the computation of the SubBytes, AES-Comb-a and AES-Comb-b will contain a mix of device-independent and device-dependent values. For instance, during round 1 (Figure 4: clock cycles 1–4) State-a will change according to the following transitions:

$$\langle I_0^0, I_1^0, I_2^0, I_3^0\rangle \to \langle D_0^1, I_1^0, I_2^0, I_3^0\rangle \to \langle D_0^1, D_1^1, I_2^0, I_3^0\rangle \to \\ \to \langle D_0^1, D_1^1, D_2^1, I_3^0\rangle \to \langle D_0^1, D_1^1, D_2^1, D_3^1\rangle$$

Such a behavior still yields a correct computation as the only value required to compute I_i^{n+1} (resp. D_i^{n+1}) during the ith clock cycle of the round computation is I_i^n (resp. D_i^n), which is still being correctly asserted by the proper state register, for all the possible selections operated by the Value Mux. In particular, the Scramble Suit architecture ensures that, at the fifth clock cycle of each round, both the device independent

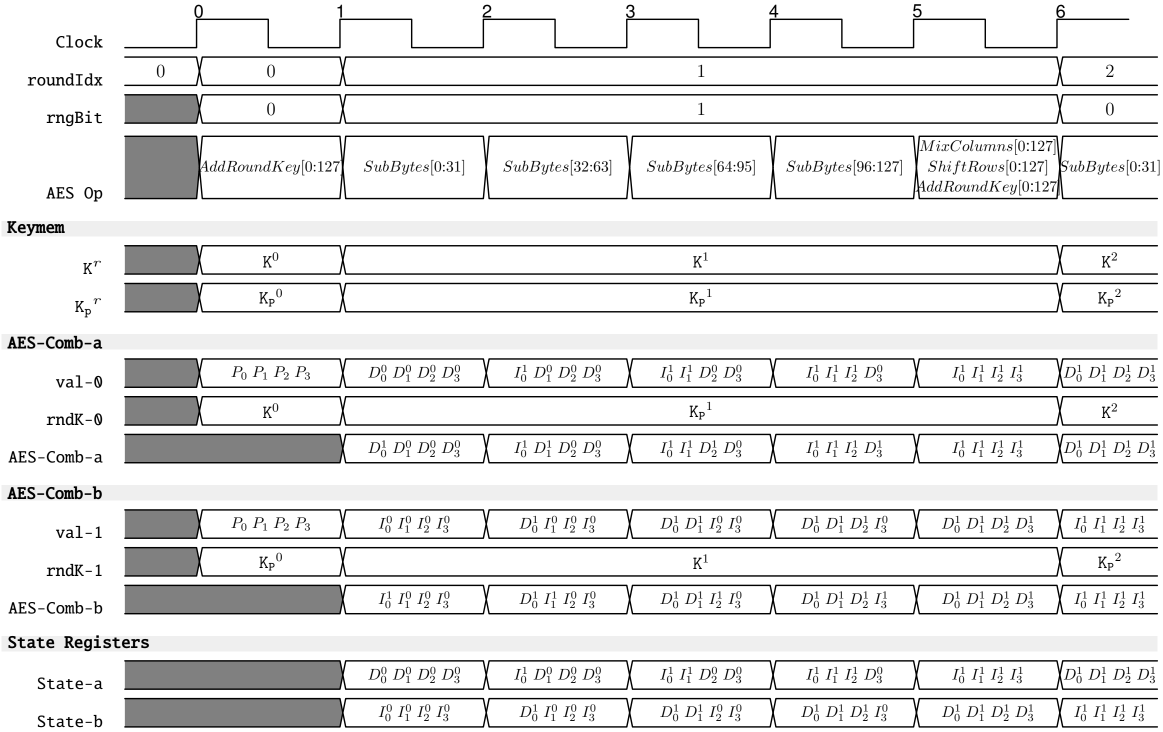


Figure 4. Timing diagram of the Scramble Suit AES-128 during the computation of round 0 and round 1, assuming a random sequence on the `rngBit` signal equal to $\langle 0, 1, 0\rangle$. The output signal of a module is labeled using the corresponding module name

state and $\mathtt{K}^r$ are muxed to the same combinatorial logic block, either `AES-Comb-a` or `AES-Comb-b`, according to the value of the `rngBit` selector, to produce the next device independent state value. The entire computation is symmetric w.r.t. the computation path employed to derive the result, and thus fulfills the assumption of non separability either in time or in space required for the Scramble Suit countermeasure. The proposed architecture retains the ability to compute a round in five clock cycles as the baseline AES core.

IV. Experimental Evaluation

We report in this section the description of the experimental setup employed to evaluate both the efficiency and the effectiveness of the Scramble Suit countermeasure applied to our case-study AES-128. In particular, we report the results of profiled attacks conducted both with Bayesian templates and SVMs on both protected and unprotected instances of the cipher deployed onto an Field Programmable Gate Array (FPGA) platform. Furthermore, we report experimental evidence maintaining the fact that the profiles of two different device instances equipped with Scramble Suit exhibit significantly different power profiles, thus preventing device profile reuse.

Experimental setup and efficiency results. We chose as our validation platform the Sakura-G development board [56], which is especially designed to allow high accuracy, low noise side channel measurements. The board is endowed with two Xilinx Spartan 6 FPGAs: the main one (XC6SLX75-2) is dedicated to the deployment of the implementation to be measured, while the other, smaller, one (XC6SLX9-2) can be employed to provide ancillary signals and control the main one. The clock is provided by an on-board, 48 MHz, quartz oscillator fed into the control FPGA and forwarded by it to the main FPGA, taking care of performing a proper buffering. We deployed the proposed AES architecture together with a Universal Asynchronous Receiver Transmitter (UART) module allowing us to both send the plaintexts and key values to the protected AES, and retrieve the result computed on the FPGA. We employed a 64-bit Linear Feedback Shift Register (LFSR) with a primitive connection polynomial to generate the `rngBit` input. We note that, although the LFSR-based Pseudo Random Number Generator (PRNG) is not cryptographically secure (as evaluating a secure random number generator is not the focus of this evaluation) it provides a uniformly distributed sequence of bits with a repetition period of $2^{64}-1$, which provides a sound countermeasure setup from a uniformity of the computation distribution standpoint. The design was synthesized with Xilinx ISE Ver. 14.7, performing a balanced area-speed synthesis.

Table I
Comparison of FPGA resource usage between a single unprotected AES core and the Scramble Suit one (excluding the PUF), synthesized for a Spartan 6 LX75 target FPGA. Area resources report both the absolute resource figure, and the percentage of the device occupied

	Baseline	**Scramble Suit**
Critical Path Slack (STA)	1.72 ns	3.70 ns
LUTs	8700 (18%)	11201 (24%)
FFs	3081 (3%)	5464 (5%)
BRAM blocks	0 (0%)	0 (0%)

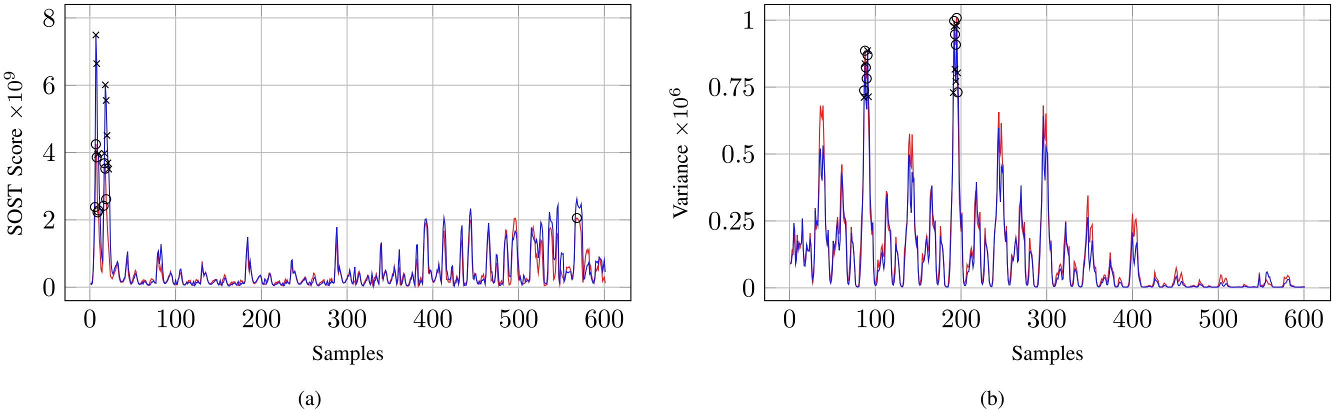


Figure 5. SOST scores (a) and variance (b) depicted as a function of the time sample, obtained from the measurements gathered from `instance-a` (blue) `instance-b` (red). The metrics were computed along the entire AES-128 computation (600 samples). All the curves are computed on the 4000 traces employed as the training set, for each virtual device instance. Both feature selection techniques identify the best time instants picking the 10 ones having the highest value of the corresponding figure of merit. Samples selected for `instance-a` are marked with a black circle, while the ones for `instance-b` are marked with a black ×

Table I reports the critical path slack according to post place-and-route Static Timing Analysis (STA) and FPGA resource utilization of both the baseline design (a single unprotected AES core) and our protected solution (without considering the PUF), showing that the Scramble Suit countermeasure requires only 28% more LUTs and 77% more flip flops, a substantial saving with respect to a straightforward implementation of our architecture, which would have implied complete resource duplication.

We note that neither implementation employs the on-die Block RAM (BRAM) blocks for the sake of an easier comparison against results obtained on different devices. We also report in the appendix of the paper a depiction of the floorplan of the placed and routed implementation of our design obtained with the Xilinx PlanAhead floorplanner.

Effectiveness validation setup. The effectiveness validation was done measuring the power consumption of two protected implementations differing only by the key employed in the AES encryption acting as PUF, thus resulting in two different virtual instances of the device running on the same silicon. Such a scenario is the ideal case for an attacker, since the only contribution to the variation of the templates is provided by Scramble Suit. In a real-world scenario, the attacker would have to deal also with the effects of process variability which will be present whenever the attacked device instances do not share the same silicon. For the sake of clarity, we will denote them as `instance-a` and `instance-b` from now on. The power traces were collected with a Picoscope 5203 Digital Sampling Oscilloscope (DSO), sampling at 500 Msamples/s, connected to the amplified SubMiniature version A (SMA) connector of the Sakura-G via two cascaded Agilent INA-10386 amplifiers providing a gain of 26 dB each, with a bandwidth of 1.5 GHz, in addition to the 14 dB at 350 MHz provided by the on-board amplifier. We employed as a trigger a dedicated signal from the protected AES core, forwarded to one of the free pins in the board pin-header, asserted at the beginning of the encryption and deasserted at its end. We took care of preventing measurement pollution from the ringing effects of the trigger assertion inserting a small delay between the trigger and the actual beginning of the encryption. All the power traces were obtained as the time-wise average of 16 measurements of the same encryption, after checking that no time-wise misalignment took place between acquisitions.

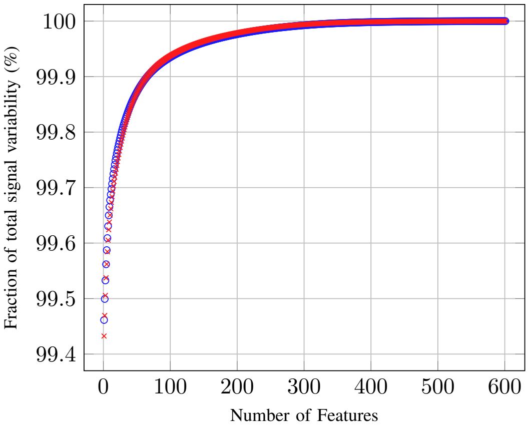
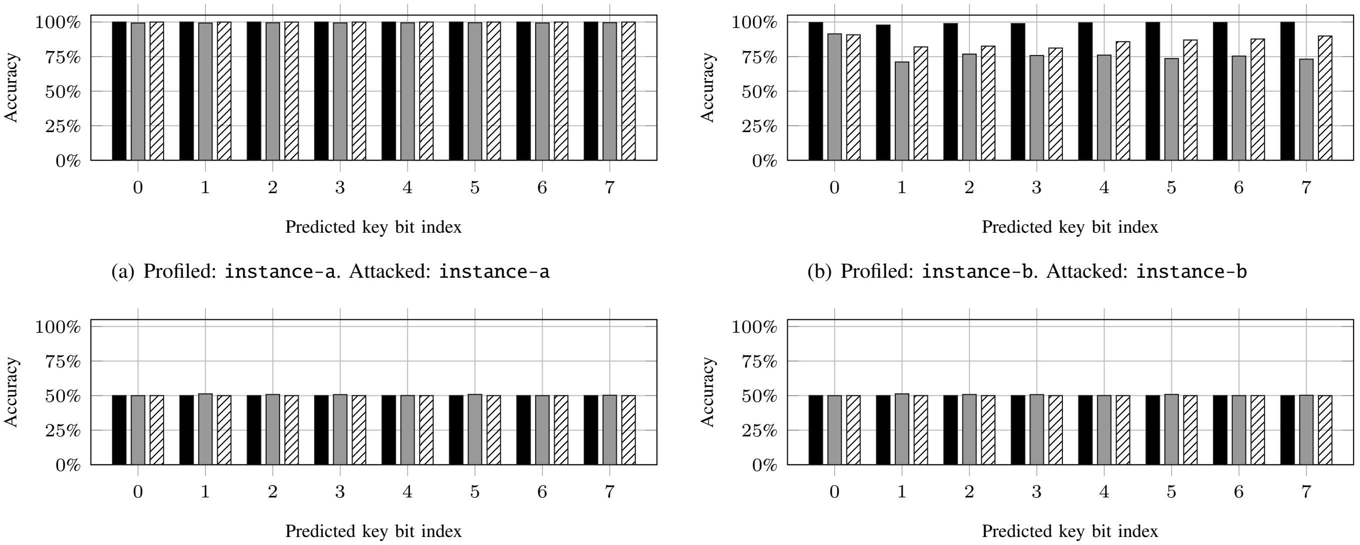


Figure 6. Fraction of the total signal variability depicted as a function of the number of principal components contributing to it. The principal components were computed by the PCA employing the 4000 traces of the training set; each trace is 600 samples long and encompasses the entire AES-128. The data obtained from `instance-a` are represented by blue circles, while the ones from `instance-b` are depicted as red ×

Experimental effectiveness validation. We validated our countermeasure against profiled attacks performed employing both Bayesian templates [19] and Support Vector Machines classifiers [21] as described in Section II. Given the fact that the number of samples s for each power trace is around 600, to avoid numerical instability and convergence problems for the classifiers, we applied the three feature selection techniques currently employed, namely the selection of trace samples with maximum variance, the selection of trace samples with the maximum Sum Of Squares of t-difference of means (SOST) [24], and the Principal Component Analysis (PCA) [25], [41], (see Section II).

Figures 5(a) and 5(b) report the values of the SOST scores and sample-wise variances computed over 4000 traces for both `instance-a` (blue) and `instance-b` (red). As it can be noticed, the figures are substantially the same between the two devices, a result which is to be expected as the points in time



(a) Profiled: `instance-a`. Attacked: `instance-a`

(b) Profiled: `instance-b`. Attacked: `instance-b`

(c) Profiled: `instance-a`. Attacked: `instance-b`

(d) Profiled: `instance-b`. Attacked: `instance-a`

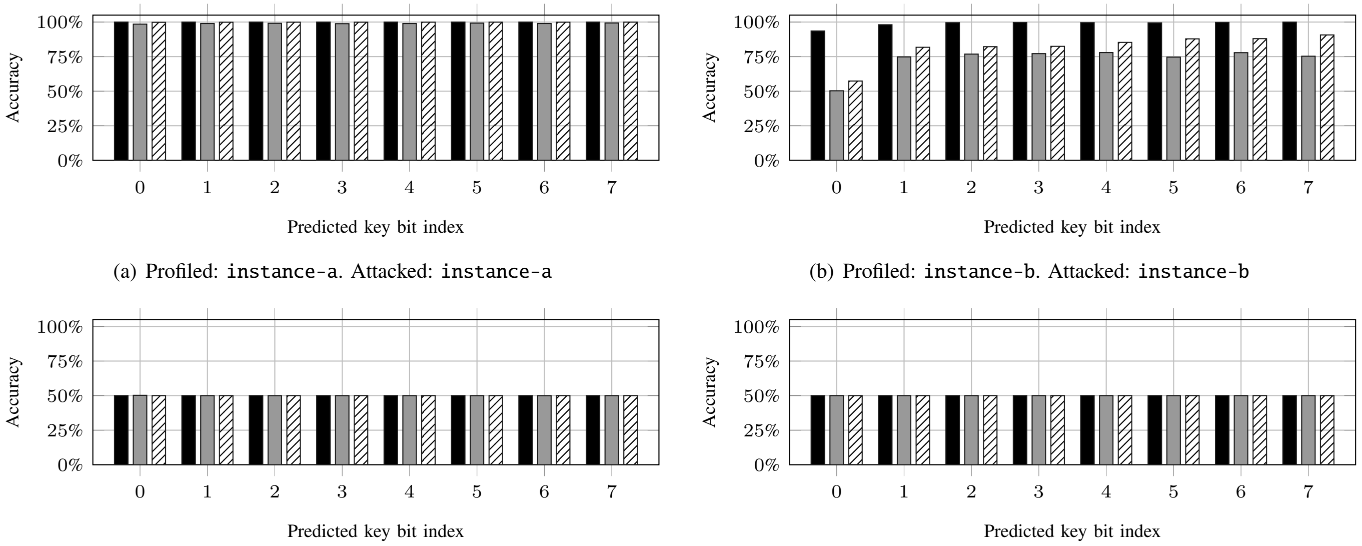
Figure 7. Template attack results. The depicted data were obtained retrieving all the key bits composing a key byte of the Scramble Suit protected AES implementation, employing Bayesian templates and PCA ▮ ▮, SOST ▯ ▯ and maximum variance ▨ ▨ feature reduction techniques. Two virtual instances of the same device, namely `instance-a`and `instance-b`, were considered. For each key bit retrieval attempt, each classifier was trained on $4{\cdot}10^3$ traces, while the accuracy percentages were computed over the classification of $40{\cdot}10^3$ traces. Subfigures 7(a) and 7(b) report the accuracy in attacking a device employing a side channel profile of the same device. Subfigures 7(c) and 7(d) report the accuracy in attacking a device exploiting the profile of the other one

exhibiting the most significant side channel leakage should be the same across different instances of the same device. We note that, while picking the points in time which exhibit the highest SOST score (i.e., the top ranked SOST features) selects time instants belonging to the first and last round of the AES-128; picking the points with maximum variance (i.e., the top ranked maximum variance features) selects time instants belonging to intermediate rounds of the cipher. The selection made by the SOST score matches the common intuition employed in non-profiled SCA, where the first rounds of the AES implementation are employed as targets.

However, the heuristic nature of the two selection criteria, that leverage the variance to select the point of interest, highlights two counter-intuitive results: *i)* the maximum variance criterion selects time instants corresponding to intermediate rounds of the AES while *ii)* SOST reports non zero scores for the time instants which match in time the last AES rounds. Finally, we note differences in time instant selection between instance-a and instance-b and the different scores in the same time instants can be ascribed to the heuristic nature of the two selection criteria which include measurement noise in the computed variance.

Figure 6 reports the fraction of total signal variability derived from the analysis of the eigenvalues derived from the feature reduction performed with the PCA, after normalizing the traces. In particular, the figure reports the percentage of the total variance taken into account as a function of how many eigenvalues are considered. We note that considering the first three eigenvalues is enough to take into account more than 99.5% of the total signal variance, while considering 10 of them we take into account 99.68% of it. We thus deemed 10 features to be sufficient to represent the information content of the traces. For the sake of simplicity in the comparison, we decided to employ the same number of features also for the SOST and maximum variance feature selection techniques (highlighted by black markers in Figures 5(a) and 5(b)).

We performed the profiled attacks training two types of classifiers on the feature-reduced traces, i.e., SVMs and Bayesian templates, to distinguish a single bit k_i of the target key byte at a time. Employing a single-bit template attack was proven to be effective, in particular against implementations having countermeasures for non-profiled attacks [57]. In particular, we trained a set of eight classifiers, i.e., one per key bit, for both `instance-a` and `instance-b`, and for each pair of feature reduction and classification technique. The SVM classifiers were realized by means of their implementation in the Matlab2017a statistic and machine learning toolbox, while we implemented from scratch the Bayesian template classifier according to the description in [19]. The SVMs were trained employing a Gaussian kernel function, and instructing the training phase to consider a maximum outliers percentage of 5%. For each bit k_i of the attacked key byte, the training set of traces for both classifiers was acquired measuring 4000 traces where $k_i{=}0$, and 4000 traces where $k_i{=}1$. The remainder of the key bits was kept to the same value, and randomly distributed plaintexts were employed. It is worth to note that the use of a higher number of traces (up to $15{\cdot}10^3$) lead to very slow, but steady, poisoning of the classifiers. On the other hand, employing 4000 traces in the training process allowed to reach a $\approx 100\%$ accuracy with both classification methods, given the use of a proper feature reduction technique. To provide an ideal scenario for the attacker, we employed the same sequence of randomly distributed plaintexts for the characterization of both `instance-a` and `instance-b`, and kept the fixed key bits which were not being classified to the same value. We evaluated the accuracy of the resulting classifiers through employing a



(a) Profiled: `instance-a`. Attacked: `instance-a`

(b) Profiled: `instance-b`. Attacked: `instance-b`

(c) Profiled: `instance-a`. Attacked: `instance-b`

(d) Profiled: `instance-b`. Attacked: `instance-a`

Figure 8. Machine learning attack results. The depicted data were obtained retrieving all the key bits composing a key byte of the Scramble Suit protected AES implementation, employing SVM classifiers and PCA ▮▮, SOST ▯▯ and maximum variance ▨▨ feature reduction techniques. Two virtual instances of the same device, namely `instance-a` and `instance-b`, were considered. For each key bit retrieval attempt, each classifier was trained on $4{\cdot}10^3$ traces, while the accuracy percentages were computed over the classification of $40{\cdot}10^3$ traces. Subfigures 8(a) and 8(b) report the accuracy in attacking a device employing a side channel profile of the same device. Subfigures 8(c) and 8(d) report the accuracy in attacking a device exploiting the profile of the other one

set of $40{\cdot}10^3$ feature-reduced traces (not in the training set), with an equal amount of traces having the classified bit set to zero or one. The $40{\cdot}10^3$ traces were collected using the same setting as the training ones, while the accuracy was computed as the percentage of traces which have been correctly labeled by the classifier.

Figure 7 depicts the results of leading a profiled attack employing eight pairs of Bayesian templates, namely eight for `instance-a` and eight for `instance-b`. The classifiers were trained to label the traces according to the value of eight bits $\{k_0, \ldots, k_7\}$ of the attacked secret key byte, employing the three different feature reduction techniques, i.e, PCA (solid black), SOST (solid gray), and maximum variance (diagonal lines). In particular, Figures 7(a) and 7(b) report the results of leading a key recovery against `instance-a`, and `instance-b` employing the classifier trained on the same device instance. The results show a minimum accuracy of 99.99% on `instance-a` and 97.80% on `instance-b`, when employing traces that have been feature-reduced with PCA.

The SOST and maximum variance feature selection techniques achieve similar results on `instance-a`, while perform worse on `instance-b`. We ascribe such a behavior to the partially heuristic nature of both the SOST and maximum variance criteria, as they both consider high variance samples to be highly informative, which may not be the case.

Figures 7(c) and 7(d) report the accuracy of performing a cross-device attack, i.e., employing the Bayesian templates obtained from the profiling of an instance to attack the other one. The results show a 50.00% maximum accuracy when employing the PCA reduction technique, which is equivalent to a random guess, and a 50.72% maximum accuracy when employing traces that have been feature-reduced with SOST. We ascribe the minimum deviation from 50.00% taking place with SOST to numerical instability as the template classifier is required to handle $40{\cdot}10^3{\times}40{\cdot}10^3$ wide matrices to compute the resulting predictions. Repeating the classification with a different set of $40{\cdot}10^3$ traces leads to smaller or no discrepancies from the ideal 50.00% accuracy.

Figure 8 reports the results of the profiled attacks performed with SVM classifiers on both `instance-a` and `instance-b`, following the same graphic conventions of Figure 7. In general the SVM classifiers perform worse than the Bayesian template ones, thus practically validating the information theoretic optimality of the latter technique in the considered scenario. In particular, machine learning attacks achieve an accuracy between 99.99% and 93.54% in identifying the correct key bit for both devices when employing traces that have been feature-reduced with PCA. Moreover, SVM classifiers are able to extract the values for the two least significant bit in `instance-b` with an accuracy of 50.28% and 74.80% when the SOST feature selection technique is employed (see Figure 8(b)), in contrast with the 91.45% and 71.10% accuracy of the Bayesian template classifiers using the same feature reduction technique (see Figure 7(b)). We ascribe the difference in the effectiveness of key extraction to the heuristic nature of the maximum variance and SOST technique, which may include samples which are characterized by a high variance, not necessarily key related. Summarizing the experimental results of the campaign, Table II reports the amount of traces required to retrieve all eight key bits querying the classifiers repeatedly and selecting the answer corresponding to the majority of the predictions. The results show how retrieving even the worst predicted bits employing the best profiled attack approach (templates and PCA) only takes 3 traces, while the random guess obtained employing a profile obtained on a different instance does not allow the key retrieval in our evaluation. Such

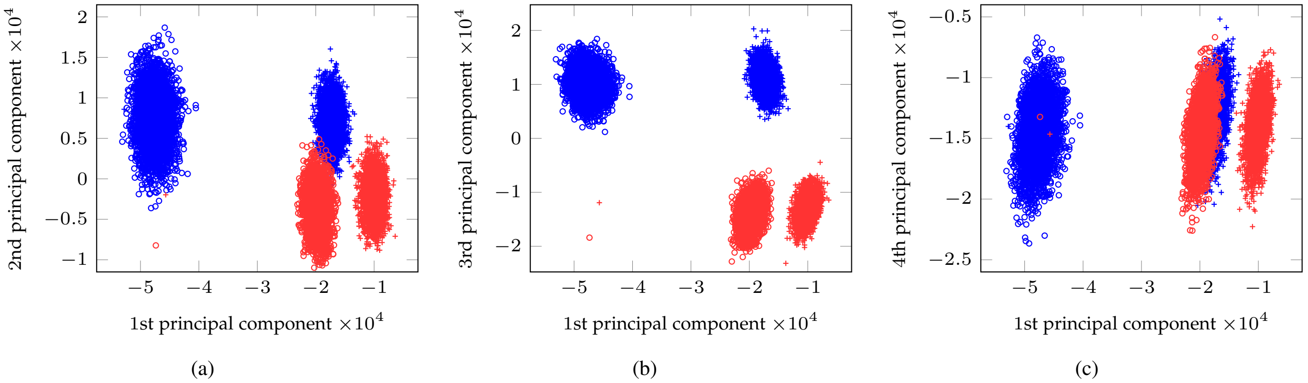


Figure 9. Depiction of the first four features computed by the PCA on each of the two sets employed to train the classifiers learning the profile of `instance-a` when the first key bit of the attacked byte is set to either 0 (circles) or 1 (crosses), respectively. The corresponding data for `instance-b` are represented with the same graphical conventions, employing the color to distinguish `instance-a` (blue) from `instance-b` (red).
The plots show that the profiles obtained from `instance-a` and `instance-b`, when the targeted key bit is set to the same value for both of them are farther apart than the profiles obtained from either one of the devices when the target key bit assumes values 0 and 1.
For the sake of clarity in visualization, the figures report three bi-dimensional projections of the data, considering respectively the 1st and 2nd principal components (a), the 1st and the 3rd (b), and the 1st and the 4th (c).
From the depicted data is possible to infer that *i)* the profile for a single device is accurate enough to lead an attack, and *ii)* it is not possible to derive a profile independent from the inter-device variability out of an averaging of the profiles coming from different device instances

Table II
Number of traces required to determine the correct value of all the key bits employing the majority of the predictions as a selection criterion, with a 99% probability of success (employing templates and PCA)

Profiled device	**Attacked device**	**Worst Accuracy**	**No. traces**
`instance-a`	`instance-a`	99.99%	3
	`instance-b`	50.00%	∞
`instance-b`	`instance-a`	50.00%	∞
	`instance-b`	97.80%	3

data point to the fact that the cross-instance misclassification is not originating by a lack of accuracy in modeling a given device instance, but instead comes from the reuse of a classifier trained for a different virtual device.

Effects of the combination of multiple templates into one. As a final point of our experimental evaluation, we consider the possibility of inferring a model for an unknown device combining the profiles obtained from multiple different ones, as reported in [30], [31]. To this end, we note that in [31] the authors analyze how to combine different profiles coming from 20 different instances of the same 65nm ASIC AES-128 implementation to derive a profile which captures non null information from the phenomenon. Employing a metric called *perceived information*, the authors conclude that combining the information from the profiles of multiple devices results in a 56% decrease of the information obtainable attacking an unknown device with respect to an attack exploiting the profile of the same device. While providing a theoretical closed-form assessment of the effects of combining multiple profiles into a single one is out of the scope of this work, we provide experimental evidence of the extent of the differences in the side channel profile of diverse instances of the same device caused by Scramble Suit. To this end, we analyze the training data employed in our classifiers after being feature-reduced via PCA. For the sake of clarity in visualization, we consider only the four dimensions with the highest eigenvalues obtained employing the PCA as a feature selection technique. We note that considering their corresponding eigenvalues in the PCA decomposition such four dimensions account for at least 99.56% of the variance of the signal in both device instances.

Figure 9 depicts the selected features for the four sets of 4000 traces, each one of which was employed to train a classifier to distinguish the value of the first bit of the key value under attack. We do not report the results for the other bits, as they are analogous and do not contribute with further information to the work. For the sake of clarity, the feature-reduced traces are depicted considering their projection on two features at once, and employing the said two features as coordinates to plot the point representing the feature-reduced trace on a bi-dimensional plane. We depict the traces coming from `instance-a` in blue and `instance-b` in red, while we represent the value of the targeted key bit employed in the measure changing the form of the mark: a circle represents a value of zero for the key bit, while a cross represents one.

First of all, we note that the profile obtained for a single device is suitable for an attack lead with a SVM classifier, as the cluster of projected traces with different key values coming from the same instance of the device can be easily separated by a straight line boundary. Moreover, Bayesian templates are also well suited to classify the device behavior, as the two populations of traces (one for each value of the key bit under attack) are well characterized by their mean and variance, as there is substantially no sample from one population within a standard-deviation-wide interval from the mean of the other.

Concerning the possibility of deriving a profile from multiple devices (two in this case), the reported data allow to infer that profiles obtained combining the traces coming from different device instances would not yield usable classifiers. In particular, we note that the distance between the two sample means of

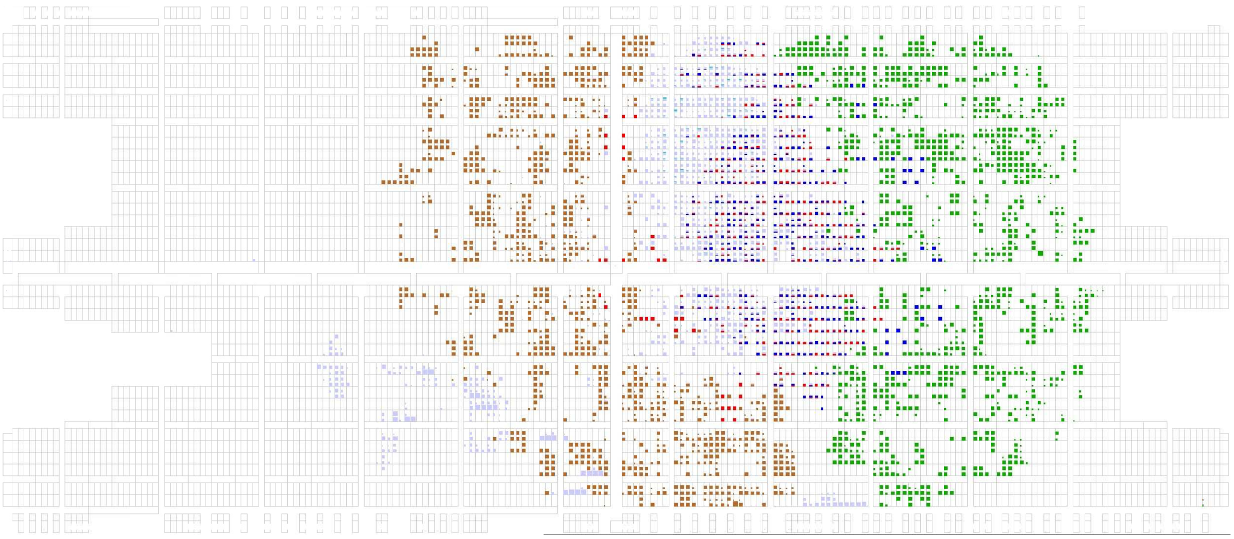


Figure 10. Floorplan of the proposed design as placed and routed on the Spartan 6 LX75 target FPGA

the sets of traces obtained with the same value of the profiled key bit on different devices is higher than the one between the set of traces obtained with different values of the key bit on the same device in almost all cases. This, in turn, implies that combining the traces obtained from different devices will result in a model showing an accurate fit for none of them, provided the sets are far enough. In this respect, we note that the Scramble Suit approach allows to further separate such sets, as adding more than one extra replica of the cipher will cause an increase in the inter-device distances among the set of traces. Indeed, adding more than one replica has the effect of pushing further the process variation amplification effect provided by Scramble Suit.

V. Concluding Remarks

We proposed Scramble Suit, an architectural countermeasure to cross-device profiled attacks that remains independent from the EDA tool employed to synthesize and to place-and-route the design. As a case study, we provided an instantiation of Scramble Suit employing an AES-128 cipher implementation and evaluated its resistance to profiled attacks against both the Bayesian template attack approach, and the machine learning approach based on SVM classifiers. In addition, three different state-of-the-art feature reduction techniques were employed to increase the effectiveness of the attacks. We proved how our countermeasure foils the key retrieval attempts via profiled attacks reducing their key derivation accuracy to a random guess, and showed an area overhead coming from our architectural design of only 28% more LUTs and 77% more flip-flops (to which the area of the chosen PUF solution should be added) with respect to an unprotected AES cipher implementation. Such figures of merit make our architectural proposal particularly interesting in case the device is already endowed with a PUF IP core for device attestation purposes.

Appendix

Figure 10 reports the floorplan of the placed and routed design of our case study protected implementation of AES-128 on the target Spartan 6 LX75 FPGA. The floorplan is obtained with Xilinx PlanAhead, which also picks the colormap to be employed to draw the floorplan. The registers holding the keyschedules for *KeyMem* and *KeyMem-P* are depicted in brown and green, respectively. The shared logic between the two implementations (e.g., `SBOX`, UART port) is depicted in grey. Components pertaining to the computation of the first datapath of AES (e.g., *AES-Comb-a*, *State-a*, read port for the `SBOX`) are depicted in red, while the corresponding ones for the second datapath are depicted in blue. It can be observed how the automated placement of the logic elements contributed to blend the circuitry composing the datapaths of the two AES instances, effectively providing a further practical hindrance to spatial separation via local EM probing. Such an effect is a welcome addition to the randomized choice of the datapath on which the two AES computations are carried out.

References

[1] E. Bertino, K. R. Choo, D. Georgakopoulos, and S. Nepal, "Internet of Things (IoT): Smart and Secure Service Delivery," *ACM Trans. Internet Techn.*, vol. 16, no. 4, pp. 22:1–22:7, 2016. [Online]. Available: http://doi.acm.org/10.1145/3013520

[2] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer, and Y. Yarom, "ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 1626–1638. [Online]. Available: http://doi.acm.org/10.1145/2976749.2978353

[3] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer US, 2007. [Online]. Available: https://doi.org/10.1007/978-0-387-38162-6

[4] D. Zoni, A. Barenghi, G. Pelosi, and W. Fornaciari, "A Comprehensive Side-Channel Information Leakage Analysis of an In-Order RISC CPU Microarchitecture," *ACM Trans. Design Autom. Electr. Syst.*, vol. 23, no. 5, pp. 57:1–57:30, 2018. [Online]. Available: https://doi.org/10.1145/3212719

[5] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "Reactive side-channel countermeasures: Applicability and quantitative security evaluation," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 62, pp. 50–60, 2018. [Online]. Available: https://doi.org/10.1016/j.micpro.2018.07.001

[6] A. Barenghi and G. Pelosi, "Side-channel security of superscalar CPUs: evaluating the impact of micro-architectural features," in *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*. ACM, 2018, pp. 120:1–120:6. [Online]. Available: https://doi.org/10.1145/3195970.3196112

[7] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "Information leakage chaff: feeding red herrings to side channel attackers," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015, pp. 33:1–33:6. [Online]. Available: https://doi.org/10.1145/2744769.2744859

[8] W. Yu, O. A. Uzun, and S. Köse, "Leveraging On-chip Voltage Regulators as a Countermeasure against Side-channel Attacks," in *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015, pp. 115:1–115:6. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744866

[9] A. Gornik, A. Moradi, J. Oehm, and C. Paar, "A Hardware-Based Countermeasure to Reduce Side-Channel Leakage: Design, Implementation, and Evaluation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1308–1319, 2015. [Online]. Available: https://doi.org/10.1109/TCAD.2015.2423274

[10] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating Masking Schemes," in *Advances in Cryptology - CRYPTO 2015, Santa Barbara, CA, USA, August 16-20, 2015*, ser. Lecture Notes in Computer Science, R. Gennaro and M. Robshaw, Eds., vol. 9215. Springer, 2015, pp. 764–783. [Online]. Available: https://doi.org/10.1007/978-3-662-47989-6_37

[11] G. Agosta, A. Barenghi, and G. Pelosi, "Compiler-based Techniques to Secure Cryptographic Embedded Software against Side Channel Attacks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 38, no. 7, pp. 1–5, 2019. [Online]. Available: https://doi.org/10.1109/TCAD.2019.2912924

[12] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "The MEET Approach: Securing Cryptographic Embedded Software Against Side Channel Attacks," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1320–1333, 2015. [Online]. Available: https://doi.org/10.1109/TCAD.2015.2430320

[13] G. Agosta, A. Barenghi, M. Maggi, and G. Pelosi, "Design space extension for secure implementation of block ciphers," *IET Computers & Digital Techniques*, vol. 8, no. 6, pp. 256–263, 2014. [Online]. Available: https://doi.org/10.1049/iet-cdt.2014.0037

[14] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "A Multiple Equivalent Execution Trace Approach to Secure Cryptographic Embedded Software," in *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*. ACM, 2014, pp. 210:1–210:6. [Online]. Available: https://doi.org/10.1145/2593069.2593073

[15] G. Agosta, A. Barenghi, and G. Pelosi, "A code morphing methodology to automate power analysis countermeasures," in *The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012*, P. Groeneveld, D. Sciuto, and S. Hassoun, Eds. ACM, 2012, pp. 77–82. [Online]. Available: https://doi.org/10.1145/2228360.2228376

[16] J. A. Ambrose, R. G. Ragel, S. Parameswaran, and A. Ignjatovic, "Multiprocessor information concealment architecture to prevent power analysis-based side channel attacks," *IET Computers & Digital Techniques*, vol. 5, no. 1, pp. 1–15, 2011. [Online]. Available: https://doi.org/10.1049/iet-cdt.2009.0097

[17] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. Springer-Verlag New York, 2012. [Online]. Available: https://doi.org/10.1007/978-1-4419-8080-9

[18] K. Tiri and I. Verbauwhede, "A Digital Design Flow for Secure Integrated Circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1197–1208, 2006. [Online]. Available: https://doi.org/10.1109/TCAD.2005.855939

[19] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," in *CHES 2002, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, B. S. K. Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523. Springer, 2002, pp. 13–28. [Online]. Available: https://doi.org/10.1007/3-540-36400-5_3

[20] G. Hospodar, B. Gierlichs, E. D. Mulder, I. Verbauwhede, and J. Vandewalle, "Machine Learning in Side-channel Analysis: A First Study," *J. Cryptographic Engineering*, vol. 1, no. 4, pp. 293–302, 2011. [Online]. Available: http://dx.doi.org/10.1007/s13389-011-0023-x

[21] L. Lerman, G. Bontempi, and O. Markowitch, "Power Analysis Attack: An Approach Based on Machine Learning," *Int. Journal of Applied Cryptography (IJACT)*, vol. 3, no. 2, pp. 97–115, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1504/IJACT.2014.062722

[22] ——, "A Machine Learning Approach against a Masked AES - Reaching the Limit of Side-channel Attacks with a Learning Model," *J. Cryptographic Engineering*, vol. 5, no. 2, pp. 123–139, 2015. [Online]. Available: http://dx.doi.org/10.1007/s13389-014-0089-3

[23] A. Heuser, O. Rioul, and S. Guilley, "Good Is Not Good Enough - Deriving Optimal Distinguishers from Communication Theory," in *CHES 2014, Busan, South Korea, Sep. 23-26, 2014.*, 2014, pp. 55–74. [Online]. Available: https://doi.org/10.1007/978-3-662-44709-3_4

[24] B. Gierlichs, K. Lemke-Rust, and C. Paar, "Templates vs. Stochastic Methods," in *CHES 2006, Yokohama, Japan, October 10-13, 2006.*, 2006, pp. 15–29. [Online]. Available: http://dx.doi.org/10.1007/11894063_2

[25] C. Archambeau, E. Peeters, F. Standaert, and J. Quisquater, "Template Attacks in Principal Subspaces," in *CHES 2006, Yokohama, Japan, October 10-13, 2006.*, 2006, pp. 1–14. [Online]. Available: https://doi.org/10.1007/11894063_1

[26] E. Oswald and S. Mangard, "Template Attacks on Masking - Resistance Is Futile," in *CT-RSA 2007, San Francisco, CA, USA, February 5-9, 2007.*, ser. Lecture Notes in Computer Science, M. Abe, Ed., vol. 4377. Springer, 2007, pp. 243–256. [Online]. Available: https://doi.org/10.1007/11967668_16

[27] W. Cheng, C. Zheng, Y. Cao, Y. Zhou, H. Zhang, S. Guilley, and L. Sauvage, "How Far Can We Reach? Breaking RSM-Masked AES-128 Implementation Using Only One Trace," Cryptology ePrint Archive, Report 2017/1144, 2017, https://eprint.iacr.org/2017/1144.

[28] V. Banciu, E. Oswald, and C. Whitnall, "Reliable information extraction for single trace attacks," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, W. Nebel and D. Atienza, Eds. ACM, 2015, pp. 133–138. [Online]. Available: http://dl.acm.org/citation.cfm?id=2755783

[29] H. L. Bouder, T. Barry, D. Couroussé, J. Lanet, and R. Lashermes, "A Template Attack Against VERIFY PIN Algorithms," in *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4: SECRYPT, Lisbon, Portugal, July 26-28, 2016.*, C. Callegari, M. van Sinderen, P. G. Sarigiannidis, P. Samarati, E. Cabello, P. Lorenz, and M. S. Obaidat, Eds. SciTePress, 2016, pp. 231–238. [Online]. Available: https://doi.org/10.5220/0005955102310238

[30] F. Standaert, T. Malkin, and M. Yung, "A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks," in *Advances in Cryptology - EUROCRYPT 2009, Cologne, Germany, April 26-30, 2009.*, ser. Lecture Notes in Computer Science, A. Joux, Ed., vol. 5479. Springer, 2009, pp. 443–461. [Online]. Available: https://doi.org/10.1007/978-3-642-01001-9_26

[31] M. Renauld, F. Standaert, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices," in *Advances in Cryptology - EUROCRYPT 2011, Tallinn, Estonia, May 15-19, 2011.*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 6632. Springer, 2011, pp. 109–128. [Online]. Available: https://doi.org/10.1007/978-3-642-20465-4_8

[32] H. Gross, S. Mangard, and T. Korak, "An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order," in *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017.*, ser. Lecture Notes in Computer Science, H. Handschuh, Ed., vol. 10159. Springer, 2017, pp. 95–112. [Online]. Available: https://doi.org/10.1007/978-3-319-52153-4_6

[33] X. Xi, A. Aysu, and M. Orshansky, "Fresh re-keying with strong PUFs: A new approach to side-channel security," in *2018 IEEE International Symp. on Hardware Oriented Security and Trust, HOST 2018, Washington, DC, USA, April 30 - May 4, 2018*. IEEE CS, 2018, pp. 118–125.

[34] C. Herder, M. M. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014. [Online]. Available: https://doi.org/10.1109/JPROC.2014.2320516

[35] C. Herder, L. Ren, M. van Dijk, M. M. Yu, and S. Devadas, "Trapdoor Computational Fuzzy Extractors and Stateless Cryptographically-Secure Physical Unclonable Functions," *IEEE Trans. Dependable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2017. [Online]. Available: https://doi.org/10.1109/TDSC.2016.2536609

[36] D. Oswald and C. Paar, "Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World," in *CHES 2011, Nara, Japan, September 28 - October 1, 2011.*, ser. Lecture Notes in Computer Science, B. Preneel and T. Takagi, Eds., vol. 6917. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-23951-9_14

[37] P. N. Fahn and P. K. Pearson, "IPA: A New Class of Power Attacks," in *CHES'99, Worcester, MA, USA, August 12-13, 1999.*, ser. Lecture Notes in Computer Science, Ç. K. Koç and C. Paar,

Eds., vol. 1717. Springer, 1999, pp. 173–186. [Online]. Available: https://doi.org/10.1007/3-540-48059-5_16

[38] M. A. Elaabid and S. Guilley, "Portability of Templates," *J. Cryptographic Engineering*, vol. 2, no. 1, pp. 63–74, 2012. [Online]. Available: https://doi.org/10.1007/s13389-012-0030-6

[39] L. Lerman, G. Bontempi, and O. Markowitch, "Side Channel Attack: an Approach based on Machine Learning," In the Proceedings of 2nd International Workshop on Constructive Side-Channel Analysis and Security Design, COSADE 2011.

[40] F. Durvaux, F. Standaert, and N. Veyrat-Charvillon, "How to Certify the Leakage of a Chip?" in *Advances in Cryptology - EUROCRYPT 2014, Copenhagen, Denmark, May 11-15, 2014.*, ser. Lecture Notes in Computer Science, P. Q. Nguyen and E. Oswald, Eds., vol. 8441. Springer, 2014, pp. 459–476. [Online]. Available: https://doi.org/10.1007/978-3-642-55220-5_26

[41] I. T. Jolliffe, "Principal Component Analysis," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Springer, 2011, pp. 1094–1096. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_455

[42] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," *Pattern Recognition*, vol. 44, no. 7, pp. 1357–1371, 2011. [Online]. Available: https://doi.org/10.1016/j.patcog.2010.12.015

[43] C. H. Chang, Y. Zheng, and L. Zhang, "A Retrospective and a Look Forward: Fifteen Years of Physical Unclonable Function Advancement," *IEEE Circuits and Systems Magazine*, vol. 17, no. 3, pp. 32–62, 2017.

[44] U. Rührmair, S. Devadas, and F. Koushanfar, *Security Based on Physical Unclonability and Disorder*. Springer New York, 2012, pp. 65–102.

[45] U. Rührmair, X. Xu, J. Sölter, A. Mahmoud, M. Majzoobi, F. Koushanfar, and W. P. Burleson, "Efficient Power and Timing Side Channels for Physical Unclonable Functions," in *CHES 2014, Busan, South Korea, Sep. 23-26, 2014.*, 2014, pp. 476–492. [Online]. Available: https://doi.org/10.1007/978-3-662-44709-3_26

[46] G. T. Becker and R. Kumar, "Active and Passive Side-Channel Attacks on Delay Based PUF Designs," *IACR Cryptology ePrint Archive*, vol. 2014, p. 287, 2014. [Online]. Available: http://eprint.iacr.org/2014/287

[47] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Controlled Physical Random Functions," in *18th Annual Computer Security Applications Conference (ACSAC 2002), 9-13 December 2002, Las Vegas, NV, USA*. IEEE CS, 2002, pp. 149–160. [Online]. Available: https://doi.org/10.1109/CSAC.2002.1176287

[48] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions," in *Towards Hardware-Intrinsic Security - Foundations and Practice*, ser. Inf. Sec. and Crypt., A. Sadeghi and D. Naccache, Eds. Springer, 2010, pp. 3–37. [Online]. Available: https://doi.org/10.1007/978-3-642-14452-3_1

[49] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," in *Advances in Cryptology - CRYPTO 2010, Santa Barbara, CA, USA, August 15-19, 2010.*, ser. Lecture Notes in Computer Science, T. Rabin, Ed., vol. 6223. Springer, 2010, pp. 631–648. [Online]. Available: https://doi.org/10.1007/978-3-642-14623-7_34

[50] R. Maes, *Physically Unclonable Functions - Constructions, Properties and Applications*. Springer, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-41395-7

[51] Joachim Strömbergson, "SecWork: Verilog Implementation of the Symmetric Block Cipher AES (Advanced Encryption Standard as specified in NIST FIPS 197," https://github.com/secworks/aes.

[52] A. Biryukov and D. Khovratovich, "Related-Key Cryptanalysis of the Full AES-192 and AES-256," in *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009.*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Springer, 2009, pp. 1–18. [Online]. Available: https://doi.org/10.1007/978-3-642-10366-7_1

[53] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*, ser. Inf. Sec. and Crypt. Springer, 2002. [Online]. Available: https://doi.org/10.1007/978-3-662-04722-4

[54] W. He, E. de la Torre, and T. Riesgo, "An Interleaved EPE-Immune PA-DPL Structure for Resisting Concentrated EM Side Channel Attacks on FPGA Implementation," in *Constructive Side-Channel Analysis and Secure Design COSADE 2012, Darmstadt, Germany, May 3-4, 2012.*, ser. Lecture Notes in Computer Science, W. Schindler and S. A. Huss, Eds., vol. 7275. Springer, 2012, pp. 39–53. [Online]. Available: https://doi.org/10.1007/978-3-642-29912-4_4

[55] L. Sauvage, S. Guilley, J. Danger, Y. Mathieu, and M. Nassar, "Successful attack on an FPGA-based WDDL DES cryptoprocessor without place and route constraints," in *Design, Automation and Test in Europe, DATE 2009, Nice, France, April 20-24, 2009*, L. Benini, G. D. Micheli, B. M. Al-Hashimi, and W. Müller, Eds. IEEE, 2009, pp. 640–645. [Online]. Available: https://doi.org/10.1109/DATE.2009.5090745

[56] H. Guntur, J. Ishii, and A. Satoh, "Side-channel AttacK User Reference Architecture board SAKURA-G," in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, 2014, pp. 271–274. [Online]. Available: https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html

[57] D. Agrawal, J. R. Rao, P. Rohatgi, and K. Schramm, "Templates as Master Keys," in *CHES 2005, Edinburgh, UK, August 29 - September 1, 2005.*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 15–29. [Online]. Available: https://doi.org/10.1007/11545262_2

[58] D. Zoni, A. Canidio, W. Fornaciari, P. Englezakis, C. Nicopoulos, and Y. Sazeides, "BlackOut: Enabling fine-grained power gating of buffers in Network-on-Chip routers," *J. Parallel Distrib. Comput.*, vol. 104, pp. 130–145, 2017. [Online]. Available: https://doi.org/10.1016/j.jpdc.2017.01.016

[59] D. Zoni, L. Cremona, and W. Fornaciari, "All-digital energy-constrained controller for general-purpose accelerators and CPUs," *IEEE embedded systems letters*, vol. 14, pp. 1–4, 2018. [Online]. Available: https://doi.org/10.1109/LES.2019.2914136

[60] D. Zoni, L. Colombo, and W. Fornaciari, "DarkCache: Energy-Performance Optimization of Tiled Multi-Cores by Adaptively Power-Gating LLC Banks," *TACO*, vol. 15, no. 2, pp. 21:1–21:26, 2018. [Online]. Available: https://doi.org/10.1145/3186895

[61] D. Zoni, L. Cremona, A. Cilardo, M. Gagliardi, and W. Fornaciari, "PowerTap: All-digital power meter modeling for run-time power monitoring," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 63, pp. 128–139, 2018. [Online]. Available: https://doi.org/10.1016/j.micpro.2018.07.007

[62] *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. ACM, 2015. [Online]. Available: http://dl.acm.org/citation.cfm?id=2744769

Alessandro Barenghi is an Assistant Professor at Politecnico di Milano, Italy. His interests include computer and network security, formal languages and compilers and he has published more than 60 papers in international peer reviewed venues.

William Fornaciari is Associate Professor at Politecnico di Milano, Italy. He published 6 books and over 200 papers in journals and conference proceedings, and holds three international patents. His research interests include embedded and cyber-physical systems, energy-aware design of sw and hw, run-time management of resources, design optimization and thermal management of multi-many cores and NoCs.

Gerardo Pelosi is an Associate Professor at Politecnico di Milano, Italy. He published more than 80 papers in journals and conference proceedings and he holds 10 patents on design of cryptographic systems. His research interests are: applied aspects of cryptography, computer security and privacy, hardware security, and secure storage and data management.

Davide Zoni is a Post-doc Researcher at the Politecnico di Milano, Italy. His research interests include RTL design and optimization for multi-cores with particular emphasis on Networks-on-Chips and coherence protocols.